B.3 Polymorphic FUN with Type Declarations

K-Annotated Syntax of Polymorphic FUN with Type Declarations

Int	::=	... all integer numbers	
Bool	::=	true | false	
Name	::=	all identifiers; to be used as names of variables, functions, and types	
Type	::=	$int \mid bool \mid unit \mid Name \mid List^{()}[Type] \rightarrow Type \mid$ ref *Type* | list *Type*	
Exp	::=	*Int* | *Bool* | *Name*	
	|	$Exp + Exp$	$[strict]$
	|	$Exp - Exp$	$[strict]$
	|	$Exp * Exp$	$[strict]$
	|	$Exp \,/\, Exp$	$[strict]$
	|	$Exp \,\%\, Exp$	$[strict]$
	|	$-Exp$	$[strict]$
	|	$Exp < Exp$	$[strict]$
	|	$Exp <= Exp$	$[strict]$
	|	$Exp > Exp$	$[strict]$
	|	$Exp >= Exp$	$[strict]$
	|	$Exp == Exp$	$[strict]$
	|	Exp != Exp	$[strict]$
	|	*Exp* and *Exp*	$[strict]$
	|	*Exp* or *Exp*	$[strict]$
	|	not *Exp*	$[strict]$
	|	fun *Params* -> *Exp*	[fun ps p -> e = fun ps -> fun p -> e]
	|	*Exp ExpList*	$[strict]$
	|	let *Binding* in *Exp*	
	|	letrec *Binding* in *Exp*	
	|	if *Exp* then *Exp* else *Exp*	$[strict]$
	|	ref *Exp*	$[strict]$
	|	$*Exp$	$[strict]$
	|	$Exp := Exp$	$[strict]$
	|	$\&Name$	$[strict]$
	|	list *ExpList*	$[strict]$
	|	list() : list(*Type*)	
	|	car *Exp*	$[strict]$
	|	cdr *Exp*	$[strict]$
	|	null? *Exp*	$[strict]$
	|	cons *Exp Exp*	$[strict]$
	|	*Exp* ; *Exp*	$[strict]$
ExpList	::=	$List^{()}\_[Exp]$	
Params	::=	$List^{()}[Name] : List^{()}[T] \mid Params, Params$	$[(xl : tl), (xl' : tl') = (xl, xl') : (tl, tl')]$
ParamsSeq	::=	*Params* | *ParamsSeq Params*	
Binding	::=	*Params* = *ExpList*	$[(ps\ p = e) = (ps =$ fun p -> $e)]$
	|	$List\_\text{and}\_[Binding]$	$[(p = el$ and $p' = el') = (p, p' = el, el')]$

K Configuration and Semantics of Polymorphic FUN with Type Declarations

$$
\begin{array}{rcl}
\mathit{KResult} & ::= & \mathit{Type} \\
\mathit{TEnv} & ::= & \mathit{Map}[\mathit{Name}, \mathit{Type}] \\
\mathit{ConfigItem} & ::= & k(K) \mid \mathit{tenv}(\mathit{TEnv}) \\
\mathit{Config} & ::= & \mathit{Type} \mid [\![K]\!] \mid [\![\mathit{Set}[\mathit{ConfigItem}]]\!]
\end{array}
$$

$[\![e]\!] = [\![k(e)\ \mathit{tenv}(\cdot)]\!]$
$[\![\langle k(t) \rangle]\!] = t$

lookup
$$\left\{ \begin{array}{l} k(\underline{\ x\ \rangle}\ \mathit{tenv}(\rho) \\ \quad \rho[x] \end{array} \right.$$

fun & app
$$\left\{ \begin{array}{l} k(\underline{\ \mathsf{fun}\ xl : tl\ \mathsf{->}\ e\ \rangle}\ \mathit{env}(\underline{\ \rho\ }) \\ \quad (tl \to e) \curvearrowright \mathit{restore}(\rho) \qquad\quad \rho[xl \leftarrow tl] \\ K ::= \cdots \mid \mathit{List}(\mathit{Type}) \to K \qquad\qquad [\mathit{strict}(2)] \\ (tl \to t)\, tl' = \mathit{applySubst}(\mathit{getSubst}(tl, tl'), t) \end{array} \right.$$

let
$$\left\{ \begin{array}{l} k(\underline{\ \mathsf{let}\ xl : tl = el\ \mathsf{in}\ e\ \rangle}\ \mathit{env}(\rho) \\ \quad \mathit{strict}(el) \curvearrowright \mathit{checkEqualTo}\ tl\ \ \mathit{andKeep} \curvearrowright \mathit{bindTo}(xl) \curvearrowright e \curvearrowright \mathit{restore}(\rho) \end{array} \right.$$

letrec
$$\left\{ \begin{array}{l} k(\underline{\ \mathsf{letrec}\ xl : tl = el\ \mathsf{in}\ e\ \rangle}\ \mathit{env}(\underline{\ \rho\ }) \\ \quad \mathit{strict}(el) \curvearrowright \mathit{checkEqualTo}\ tl\ \ \mathit{andDiscard} \curvearrowright e \curvearrowright \mathit{restore}(\rho) \qquad \rho[xl \leftarrow tl] \end{array} \right.$$

ord ops
$$\left\{ \begin{array}{l} \mathit{int} + \mathit{int} \to \mathit{int},\ \ \mathit{int} - \mathit{int} \to \mathit{int},\ \ \mathit{int}\ *\ \mathit{int} \to \mathit{int}, \\ \mathit{int}\ /\ \mathit{int} \to \mathit{int},\ \ \mathit{int}\ \%\ \mathit{int} \to \mathit{int},\ \ \ -\mathit{int}\ \to \mathit{int}, \\ \mathit{int} < \mathit{int} \to \mathit{bool},\ \ \mathit{int} <= \mathit{int} \to \mathit{bool},\ \ \mathit{int} > \mathit{int} \to \mathit{bool}, \\ \mathit{int} >= \mathit{int} \to \mathit{bool},\ \ \mathit{int} == \mathit{int} \to \mathit{bool},\ \ \mathit{int}\ \mathsf{!=}\ \mathit{int} \to \mathit{bool}, \\ \mathit{bool}\ \mathsf{and}\ \mathit{bool} \to \mathit{bool},\ \ \mathit{bool}\ \mathsf{or}\ \mathit{bool} \to \mathit{bool},\ \ \mathsf{not}\ \mathit{bool} \to \mathit{bool} \end{array} \right.$$

references
$$\left\{ \begin{array}{l} \&\ t \to \mathsf{ref}\ t \\ *\ \mathsf{ref}\ t \to t \\ \mathsf{ref}\ t := t \to \mathit{unit} \end{array} \right.$$

conditional
$$\left\{ \ \mathsf{if\ true\ \ then}\ t\ \ \mathsf{else}\ t \to t \right.$$

lists
$$\left\{ \begin{array}{l} \mathsf{list}() : (\mathsf{list}(t)) \to \mathsf{list}(t) \\ \mathsf{list}(t, t, tl) \to \mathsf{list}(t, tl) \\ \mathsf{car\ list}(t) \to t \\ \mathsf{cdr\ list}(t) \to \mathsf{list}(t) \\ \mathsf{null?\ list}(t) \to \mathit{bool} \\ \mathsf{cons}\ t\ \mathsf{list}(t) \to \mathsf{list}(t) \end{array} \right.$$

seq comp
$$\left\{ \ \mathit{unit}; t \to t \right.$$